Better Knowledge Management through Knowledge Engineering

Alun Preece, Alan Flett, and Derek Sleeman, *University of Aberdeen*
David Curry, Nigel Meany, and Phil Perry, *Baker Hughes OASIS*

Currently, few organizations have a systematic process for capturing knowledge, as distinct from data. The authors illustrate how a large oil and gas service company uses knowledge-engineering processes to capture, store, and deploy drilling-optimization knowledge.

In recent years, *knowledge management* has referred to efforts to capture, store, and deploy knowledge using a combination of information technology and business processes.[1–3] More specifically, organizations aim to acquire knowledge from valued individuals and to analyze business activities to learn from successes and failures. Such captured knowledge must then be made available throughout the organization in a timely manner.

In terms of technology, most current knowledge management activities rely on database and Internet systems. If knowledge is stored explicitly at all, it is typically in databases either as simple tables (for example, relational databases) or semistructured text (as in Lotus Notes). The use of sophisticated knowledge representation systems such as Classic, Loom, or G2 is rare. Also, few organizations have a systematic process for capturing knowledge, as distinct from capturing information. (See the "Current Practice" sidebar for a description of techniques.)

We believe that current knowledge management practice significantly under-utilizes knowledge-engineering technology, despite recent efforts to promote its use.[4] In this article, we focus on two knowledge-engineering processes:

- using knowledge acquisition processes to capture structured knowledge systematically and
- using knowledge representation technology to store the knowledge, preserving important relationships that are far richer than those possible in conventional databases.

To demonstrate the usefulness of these processes, we present a case study in which the drilling optimization group of a large oil and gas service company uses knowledge-engineering practices to support the three facets of the knowledge management task:

- *Knowledge capture*—In the group's systematic knowledge acquisition process, a conceptual business model of the company guides case and rule capture.
- *Knowledge storage*—The group uses a knowledge representation language to codify the structured knowledge in several knowledge bases, which together make up a knowledge repository.
- *Knowledge deployment*—Through standard Web browsers on the company intranet, group members can run the knowledge bases within a knowledge server. The server answers queries far more complex than those possible with conventional database systems.

Applying knowledge engineering to knowledge management

In the 1990s, knowledge engineering emerged as a mature field, distinct from but closely related to software engineering.[3,5] Among its distinct aspects are a range of techniques for knowledge elicitation and modeling, a collection of formalisms for representing knowledge, and a toolkit of mechanisms for implementing automated reasoning.

1094-7167/01/$10.00 © 2001 IEEE

Exhibit C

Here is an outline of the knowledge-engineering process:[3,6]

1. *Requirements analysis.* Identify the scope of the knowledge-based system, typically in terms of its expected competency (for example, the kinds of queries it will be able to answer).
2. *Conceptual modeling.* Based on the scope defined in step 1, create a glossary of terminology (concepts) for the application domain and define interrelationships between the terms of and constraints on their usage. An explicit conceptual model of this kind is commonly called an *ontology*.
3. *Knowledge base construction.* Using the conceptual model or ontology from step 2 as a collection of *knowledge containers* (or schemata), populate the knowledge base with *instances* of domain knowledge (often in the form of rules, facts, cases, or constraints).
4. *Operationalization and validation.* Operationalize the knowledge base from step 3 using automated reasoning mechanisms and validate its competence against the requirements from step 1. If satisfactory, release the system; otherwise, repeat steps 1 through 4 until satisfactory.
5. *Refinement and maintenance.* After delivery, the system continues to evolve as knowledge changes. Thus, steps 1 through 4 must be repeated throughout the life of the system.

Any knowledge management system that involves explicit knowledge representation is amenable to development using at least part of this process. In fact, it is always worth applying at least part of this process to any knowledge management activity that involves explicit knowledge representation. Here are several examples, using the common knowledge management activities described in the "Current Practice" sidebar:

- *Document management systems.* As a minimum, apply step 1 at the outset to ensure competency criteria are defined. This ensures at least the selection of the right tool; it may reveal a need for a more structured approach.
- *Discussion forums.* As a minimum, apply steps 1 and 2 to ensure that the system's scope is well understood, and that each forum's organization effectively supports existing (or desired) communities of practice.
- *Capability management systems.* As above, apply steps 1 and 2 to define the metaknowledge that will serve as knowledge containers or schemata to capture workers' capabilities. Use step 3 to populate the CV database.
- *Lessons-learned knowledge base systems.* Because these are knowledge-based systems, they should follow the entire five-stage process.

It is particularly important to employ knowledge-engineering techniques when an organization employs a range of knowledge management approaches. This is becoming common in larger organizations, which already use a multiplicity of information systems tied into an intranet and see a multifaceted knowledge management system as normal. For example, such a knowledge management system might include a capability management system, discussion forums, a document management system, and several lessons-learned knowledge bases. In such cases, the key challenge becomes *knowledge integration*—linking the various sources at the knowledge-content level.

In this context, the organization can use the knowledge-engineering process to define an organizational knowledge model—a *knowledge map*[7]—which delineates the relationships that bind the multifaceted knowledge management system at the knowledge-content level. (The actual software-level bindings can use hyperlinking, remote procedure calling, or any one of a host of distributed computing techniques.) Therefore, even when an organization embarks on its first, single-facet knowledge management project, it may well be worthwhile to follow steps 1 and 2 of the knowledge-engineering process to define an initial knowledge map.

Case study: drilling optimization

Baker Hughes OASIS, an engineering services subsidiary company of Baker Hughes, provides drilling-process expertise in the oil and gas industry worldwide. In particular,

Current Practice

Most knowledge management activities combine business processes and information technology.[1] As currently practiced, knowledge management includes several activities and technologies:

- *Document management systems* allow workers to find existing documents relevant to the task at hand. Essentially, these are multisource search and information-retrieval systems that tie into an organization's intranet (and may extend to the public Internet). These systems include several commercially available products, such as those made by Autonomy and Verity.
- *Discussion forum systems* promote knowledge dissemination within communities of practice. Workers subscribe to forums relevant to their interests, exchanging questions and answers, lessons learned, announcements, and industry gossip. Such systems are easily implementable with both freely available Web software and commercial products.
- *Capability management systems* allow an organization to "know who knows what."[2] Essentially, these are databases of suitably structured CVs or resumes; as such, they are implementable with off-the-shelf database software. The goal is to put people together by matching one person's need for expertise with another person's listed skills.
- *Lessons-learned knowledge base systems* let workers tap into past experience, by storing that experience as structured cases. These systems allow sophisticated queries, typically supporting "fuzzy" retrieval of "similar" cases. Although simple systems can use just conventional database software, full functionality requires special-purpose, case-based reasoning or knowledge-based system software.

References

1. W. Bukowitz and R. Williams, *Knowledge Management Fieldbook*, Prentice-Hall, Old Tappan, N.J., 1999.

2. J. Stader and A. Macintosh, "Capability Modeling and Knowledge Management," *Applications and Innovations in Intelligent Systems VII*, Springer-Verlag, Berlin, 1999, pp. 33–50.

Baker Hughes OASIS specializes in *drilling performance optimization*, which involves identifying, understanding, and overcoming barriers to improved drilling performance. Drilling performance optimization engineers need a specialized set of skills, which they draw from mechanical engineering, geology, physics, and other disciplines. Because the field is relatively new, the community of skilled optimization engineers is small, and those within Baker Hughes OASIS are dispersed worldwide.

For these reasons, drilling performance optimization represents an ideal application domain for knowledge management. Having recognized this in the early 1990s, Baker Hughes OASIS developed a multifaceted knowledge management approach, which currently includes the following system components:

- *Drilling Performance Guidelines,* a semistructured document base implemented in Lotus Notes/Domino;[8]
- *OASIS University*, an online training system for optimization engineers, also implemented in Lotus Notes/Domino;
- *Drill Bit Advisor*, a rule-based expert system implemented in LISP/CLOS using a custom graphical rule representation;[9] and
- *Drilling Knowledge Store,* a technical lessons-learned knowledge base.

All of these components are interlinked. For example, a conclusion (recommendation) made by the Drill Bit Advisor is commonly linked with a URL to a Drilling Performance Guideline in the Lotus Notes/Domino system.

The Drilling Knowledge Store, one of the newest components of this knowledge management strategy, is an open repository of case-based drilling knowledge, accessed through a Lotus Domino server. A structured search tool allows users to query the knowledge store for lessons learned in environments similar to a specified environment of interest. New knowledge forms promote easy entry of new cases, which the system submits to reviewers for audit and approval before making them available to other users. Links to the Drilling Performance Guidelines system avoid knowledge duplication and case updating and maintenance.

The Drilling Knowledge Store builds on a knowledge map developed using the standard knowledge-engineering process described earlier, and it incorporates a drilling knowledge repository, a case-base of optimization engineers' documented experience. The drilling optimization group developed this case-base in collaboration with the University of Aberdeen, managing the work as a Teaching Company Scheme. The following sections detail its development stages.

Requirements analysis

The development team first conducted a series of interviews with optimization engineers to explore the scope of the drilling knowledge repository. The key finding was that the system ought to be highly open. Because drilling optimization is relatively new, knowledge in the domain is evolving. As a result, the system would most likely have to cope with the following kinds of change:

> By the time the team had defined a reasonably complete conceptual model, they had already elicited several sample cases from the optimization engineers as a natural part of exploring the scope of the domain.

- New concepts and relationships could be discovered in the future, so knowledge containers or schemata would have to be highly extensible.
- New cases would grow in proportion to the growth in the drilling optimization business, so instances would frequently be added.
- Instances might be reclassified, especially as outdated knowledge is "decommissioned."

Conceptual modeling

Following the first round of interviewing, the development team drew up an initial glossary of terms. In an attempt to derive a set of concepts, the team analyzed the transcripts of the interviews using the PC-PACK[4] knowledge acquisition software toolkit. However, it was not sufficiently flexible in dealing with concepts where the "defining" words were not adjacent in a piece of text or where they were interspersed with words from other concepts. PC-PACK and similar textual mark-up systems allow the user to indicate only that single words correspond to concepts, attributes, and values. In practice, such entities are often defined by several words, and these are not necessarily adjacent. For example, the text "a bus system that links all the suburbs to the center and to each other" contains the concept *comprehensive-city-bus-network*, but it also contains parts of the concept *city* (suburb and center).

In view of this tool's limitations, the team used a manual concept-mapping approach instead,[10] which focused on defining concepts in two areas:

- concepts associated with the drilling environment, including extensive definitions of geological concepts (leading to the creation of an ontology for representing the rock formations that constitute a drilling task), and those associated with drilling itself (chiefly drill bits, fluids, and related apparatus); and
- knowledge management concepts that would allow the capture of useful instances of the optimization engineers' experience (most obviously, the concept of a *case*).

Early in the process, the team formalized these concepts to manage them within a software environment. They chose the Loom knowledge representation system[11] and its associated Ontosaurus browser/editor because it had a number of advantages. First, Loom is one of the most flexible and least constraining knowledge representation systems available. In addition, Loom's operational mechanisms (chiefly the classification engine) allowed the knowledge-engineering team to test the conceptual model's integrity during its development. Finally, Ontosaurus provides a Web front end for Loom knowledge bases, allowing multiple users to inspect, query, and modify the knowledge base on a network, using a standard Web browser.

Knowledge base construction

By the time the knowledge-engineering team had defined a reasonably complete conceptual model, they had already elicited several sample cases from the optimization engineers as a natural part of exploring the scope of the domain. When formalizing the conceptual model in Loom, the team took the opportunity to represent these cases using

Loom's knowledge containers. However, they used a distinct approach for systematic case acquisition.

First, they identified a small number of high-performing, expert optimization engineers. Then, team members conducted intensive, one-on-one *knowledge acquisition campaigns* with these individuals. Mindful of lessons learned from negative knowledge acquisition experiences during the heyday of expert systems in the 1980s, the team carefully designed these campaigns to ensure that the experts would contribute actively and positively.

The team formalized the knowledge acquired from each campaign in Loom, but also wrote it up in a natural-language electronic document called a *knowledge book*,[12] which the expert could check for accuracy and which was disseminated on CD-ROM throughout the company as an easily accessible, early result of the OASIS group's work.

Operationalization and validation

The knowledge base was operationalized naturally by the choice of Loom as the representation language. The team performed validation of the represented knowledge at two levels—indirect validation using the knowledge books and direct validation using Loom's inference mechanisms. Further indirect validation came through the development of drill bit selection rules using some of the case-based knowledge acquired in the knowledge acquisition campaigns. These rules were then validated using existing software in the Drill Bit Advisor expert system.

The drilling knowledge repository in Loom

Philosophically, we consider OASIS a knowledge storage and retrieval system rather than a knowledge-based system. This is because knowledge-based systems are strongly associated with tasks, such as decision support, automated decision-making, or training. But the task for which knowledge is to be used places a strong bias on the form of the knowledge.[13] In this case, the implemented system had to be task-neutral: it was to serve purely as a repository for captured knowledge, without any risk of biasing the form and content of the knowledge toward a particular future usage.

Nevertheless, the system does have at its core a set of automatic deductive facilities (provided by Loom), which operate on the definitions given to the system by the knowledge modeler. However, these inferences operate at the conceptual-model level and not at any action level. Thus, actions are left to humans, and the system does not advise per se on any action the user should take. For example, the system can recognize instances and classify them appropriately with reference to the conceptual model, but this is purely for the purpose of retrieving those instances and bringing them to the user's attention.

The Loom knowledge store has two main parts—a conceptual model and a database. (This is analogous to a database, with its schema and data parts.) The conceptual part of the knowledge base is defined using concepts. It includes binary concepts (known as *roles*) and unary concepts (known as *concepts*). The database is populated with instances of these concepts.

The system does have at its core a set of automatic deductive facilities [provided by Loom], which operate on the definitions given to the system by the knowledge modeler.

The following sections give examples of the Loom constructs to illustrate the approach in concrete terms. Our intention is to explain the constructs so that a full understanding of the representation language is not necessary. (For readers unfamiliar with Loom or similar languages, Robert MacGregor and Ronald J. Brachman and his colleagues provide good introductions.[11,14])

Modeling constructs for drilling engineers' experience

Because the knowledge store is chiefly intended to capture experiential cases from drilling engineers, the most important concept is the *case*.

```
(defconcept CASE :is-primitive
    (:and (:exactly 1 formation-sequence)
        (:all decision DECISION)
        (:all observation OBSERVATION)))
```

A case usually describes a *drill bit run*—a continuous period of drilling with a single drill bit. So, if an optimization engineer experiences some bit run worthy of being recorded in the knowledge store, the engineer should include a representation of the rock formation sequence and the decisions made on how to drill that formation sequence, along with any associated observations. A decision can refer to a choice of drill bit, mud (drilling fluid), flow rate, and so on. Alternatively, the case need not refer to an actual drill bit run if the person entering it simply has an experience to share.

A *decision* has several different dimensions, including *issues*, *actions*, *goals*, an author, a *spin*, and *reasoning*. These dimensions provide a balance between structured knowledge and free text. The structured knowledge enables formal representation and therefore supports powerful searches; the free text supports semistructured knowledge.

```
(defconcept DECISION :is-primitive
    (:and (:exactly 1 action)
        (:at-most 10 issue)
        (:at-most 10 goal)
        (:at-most 1 authors-reasoning)
        (:at-most 1 companys-reasoning)
        (:at-most 1 author)
        (:at-most 1 spin)))
```

An *issue* is some informational context that the engineer considered when making the decision. The issues in the current knowledge base reflect quite strongly the best-practice drilling database (in Lotus Notes), as shown by the link roles in the following code. These can be filled with links to other media, including the Notes database itself, using URLs.

```
(defconcept ISSUE :is-primitive
    (:and KNOWLEDGE_MANAGEMENT_CONCEPT
        (:at-most 1 symptoms-and-diagnosis-link)
        (:at-most 1 description-link)
        (:at-most 1 parameters-link)
        (:at-most 1 diagnostic-information-link)
        (:at-most 1 planning-actions-link)
        (:at-most 1 operating-practices-link)
        (:at-most 1 examples-link)))
```

An *action* is the real-world consequent the engineer performed as part of the decision; this includes both structured (categorical-outcome) and free text (textual-outcome) outcomes.

```
(defconcept ACTION :is-primitive
    (:and KNOWLEOGE_MANAGEMENT_CONCEPT
        (:at-most 1 categorical-outcome)
        (:at-most 1 textual-outcome)))
```

The system captures two kinds of *reasoning* for a decision. *The author's reasoning* is a free-text field for explanations—for example, why an engineer chose a certain drill bit. This allows the storage of incomplete, inaccurate, and even incoherent explanations for actions. After all, the main reasoning or determinism for the action consists of the other structured information describing the circumstances for the action, such as the formation sequence. The *company's reasoning* field expresses the company's commonly agreed on beliefs for the decision in question.

Modeling constructs for the drilling environment

The system describes the drilling environment chiefly in terms of conceptual rock sequences. The team achieved representations of these by defining an ontology of geological concepts, including constraints. For instance, if the user wishes to specify the depth or length of a particular section of *lithology* (a basic rock type—for example, sand or shale), that section must be represented as a *formation*. The superstructure larger than that is the *formation sequence*, which can have one or more *formations*. Each formation can have one or more *lithologies*. A formation is the conceptual modeling granularity at which the users should represent any part of the wells they feel should have represented interval lengths and depths.

```
(defconcept FORMATION_SEQUENCE :is-primitive
    (:and ROCK_CONCEPT
        (:at-least 1 formation)))

(defconcept FORMATION :is-primitive
    (:and ROCK_CONCEPT
        (:at-least 1 lithology)))
```

To allow users to represent and query formation sequences flexibly, the ontology defines several relations. For example, the relation comes-in-somewhere-after relates two formations, the first of which comes in somewhere after the other.

```
(defrelation comes-in-somewhere-after
  :domain FORMATION_SEQUENCE
  :range FORMATION
  :characteristics (:multiple-valued :closed-world)
  :is (:satisfies (?formation-x ?formation-y)
    (:and
      (FORMATION ?formation-x)
      (FORMATION ?formation-y)
      (:or (comes-in-immediately-after
            ?formation-x ?formation-y)
        (:exists (?formation-z)
          (:and
            (FORMATION ?formation-z)
            (comes-in-somewhere-after
              ?formation-x ?formation-z)
            (comes-in-somewhere-after ?formation-z
              ?formation-y)))))))
```

One important feature of lithologies is their *hardness*. While a lithology has, by definition, one rock type (such as shale), it can have more than one hardness. (For example, shale could consist of 100 meters of *very soft rock* and 300 meters of *soft rock*.)

The system describes the drilling environment chiefly in terms of conceptual rock sequences. The team achieved representations of these by defining an ontology of geological concepts.

```
(defrelation hardness
    :domain LITHOLOGY
    :range HARDNESS
    :characteristics (:closed-world :multiple-valued))
```

To support drill bit run modeling, the ontology includes a collection of functions that relate formation sequences, constituent lithologies, and accumulated hardness.

In addition to the generic geological concepts, the knowledge store includes representations of the concepts involved in drilling, such as *drill bit*.

```
(defconcept DRILL_BIT :is-primitive
    (:and DOWN-HOLE_EQUIPMENT_CONCEPT
        (:exactly 1 bit-gauge)))
```

Querying the knowledge store

The retrieve function, which retrieves instances from the knowledge base, provides an interface to Loom's deductive query facility. Formation sequence queries are among the most sophisticated forms of query that users can issue to the knowledge store. The concepts likely to be of interest are individual formations and formation sequences. Two common queries are on an overall cumulative amount of a certain hardness of a particular lithology over a formation sequence and formations that have amounts of particular lithologies of a certain hardness.

The following example query looks for cases that have a formation sequence that has as constituents of its formation(s) at least 1,900 feet of very soft to soft shale (including all subtypes of shale).

```
(retrieve ?case
  (:and
    (CASE ?case)
    (>= (sum (:collect ?lithology-amount-ft
      (:and
        (:exists (?formation-sequence ?formation
            ?lithology ?hardness)
          (:and
            (formation-sequence ?case ?formation-
              sequence)
            (formation ?formation-sequence
              ?formation)
            (lithology ?formation ?lithology)
            (lithology-hardness-amount-ft ?lithology
              ?hardness ?lithology-amount-ft)
            (:or
              (VERY_SOFT ?hardness)
              (SOFT ?hardness))
            (SHALE ?lithology)
            ))))) 1900)))
```

Users typically also want to look for cases in which engineers achieved specific goals or outcomes. The following example query retrieves cases that have a drill bit decision in which one of its goals was *good ROP* (rate of penetration) with *good bit cleaning*.

```
(retrieve ?case
  (:and
    (CASE ?case)
    (:exists (?decision)
      (decision ?case ?decision)
      (DRILL_BIT_PLANNING_DECISION ?decision)
      (goal ?decision
GOOD_ROP_WITH_GOOD_BIT_CLEANING))))
```

It is worth emphasizing how the Loom representation supports querying. First, the classification engine automatically associates new concepts (including new cases) with

super- and subconcepts. This means that a query for a subconcept will automatically find all superconcepts. This also means that, by using the Ontosaurus concept browser, a user can quickly find subconcepts related to the result of a query. Second, the pattern-matching mechanism, combined with the way Loom represents drilling sequences, means that the system easily accommodates partial matches. A user need only specify discontinuous fragments of a formation sequence, for example, to retrieve useful cases of drilling wells including those sequence fragments.

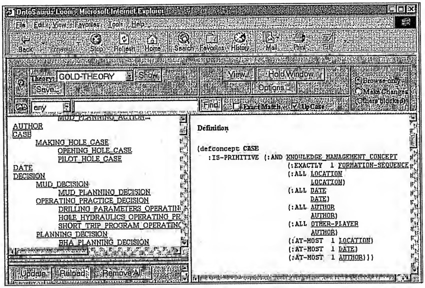

Figure 1. Loom Drilling Knowledge Repository screenshot.

Adding to the knowledge store

As we described earlier, the knowledge store comprises a conceptual and a database part. We consider the conceptual part stable and expect that knowledge will rarely need to be added or modified. However, additions to the database part will surely be regular. The Loom operations used to update the database part of the knowledge base are tell and about: tell is used to assert propositions and facts about the world or domain; about references the instance to which those propositions refer. The following example shows how a user might enter a case instance. This example case has one formation sequence name and zero or more decisions and observations.

```
(tell (:about Case-Name
    CASE
    (formation-sequence Formation-Sequence-
        Name)
    (decision Decision-Name)
    (observation Observation-Name)))
```

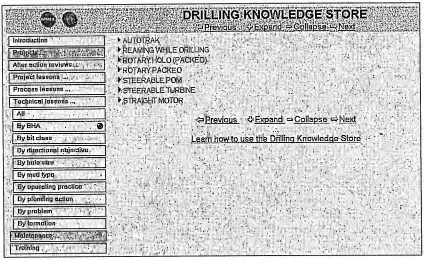

Figure 2. Lotus Notes/Domino drilling knowledge store screenshot.

Current status and future plans

The Loom Drilling Knowledge Repository currently contains 1,200 concepts and 240 relations, with further expansion planned. The knowledge store is accessible on the company's intranet using a standard Web browser through the Ontosaurus system (see Figure 1).

The use of Loom has facilitated great flexibility in the modeling process allowing the ontology to grow naturally over the first two years of the project. Attempting to model a comparable richness of interrelationships in a relational database, for example, would have been extremely difficult and more time-consuming, and it would doubtless have involved many more modifications to the schemas.

Nevertheless, although it is relatively straightforward to browse the case base and ontology using Ontosaurus, the current system has several significant problems:

- It is difficult to issue complex queries to the Loom knowledge store through the Ontosaurus interface because Ontosaurus provides direct support only for simple queries (retrieve cases with matching simple role values).
- It is hard to add new cases because these require the user to have knowledge of Loom syntax, an unrealistic expectation for optimization engineers.
- Multiuser access (basic locking and restricted concurrent access) is limited.

In addition to these issues, the team wanted the drilling knowledge repository to have a familiar interface, preferably that of the existing systems implemented using Lotus Notes/Domino. At the same time, the team wanted to link the knowledge represented in the Loom repository with information and knowledge relating to the performance optimization projects that yielded the stored knowledge. Thus, the team went with an interim solution, partially incorporating the Loom knowledge map and cases into a Lotus Notes/Domino database of project-related knowledge, to provide structure for technical lessons learned on each project. Figure 2 is a screenshot of the ported system. The immediate benefits of this included

able for knowledge verification and integrity checking.

In doing this work, we detected two weaknesses in current knowledge-engineering techniques and technology. First, as we noted, PC-PACK and other textual mark-up systems do not cope adequately with concepts defined by several nonadjacent words. Thus, we have identified the need for a more flexible tool. Second, it is very difficult to integrate expressive reasoning tools such as Loom with intranet knowledge management environments such as Lotus Notes/Domino. It seems reasonable to conclude, therefore, that while knowledge-engineering processes are ready to bring significant benefits to knowledge management projects, the knowledge-engineering toolbox needs some improvement.

Acknowledgments

The case study described here was supported by Teaching Company Scheme funding from the UK Department of Trade and Industry and the Engineering and Physical Sciences Research Council. The Lotus Notes version of the Drilling Knowledge Store was constructed largely by John Lawton and David Bowden, Transition Associates, who also proposed and developed the OASIS University. The authors thank Tom Russ of ISI/University of Southern California for advice and technical assistance regarding Loom and Ontosaurus. This article is published with the kind consent of the Hughes Christensen Company.

References

1. *Harvard Business Review on Knowledge Management*, Harvard Business School Press, Cambridge, Mass., 1998.
2. J. Liebowitz and L. Wilcox, *Knowledge Management and Its Integrative Elements*, CRC Press, Boca Raton, Fla., 1997.
3. G. Schreiber et al., *Knowledge Engineering and Management*, MIT Press, Cambridge, Mass., 2000.
4. N. Milton et al., "Towards a Knowledge Technology for Knowledge Management," *Int'l J. Human-Computer Studies*, vol. 51, no. 3, 1999, pp. 615–641.
5. E. Motta, *Reusable Components for Knowledge Modeling*, IOS Press, Amsterdam, 1999.
6. S. Russell and P. Norvig, *Artificial Intelligence*, Prentice-Hall, Old Tappan, N.J., 1995.
7. J. Domingue and E. Motta, "Planet-Onto: From News Publishing to Integrated Knowledge Management Support," *IEEE Intelligent Systems*, May/June, 2000, pp. 26–32.
8. D.A. Curry, A.V. Singelstad, and D. Bowden, "Drilling Performance Guidelines—A Tool for Sharing Drilling-Related Knowledge and Experience," SPE/IADC paper no. 52804, *SPE/IADC Drilling Conf.*, Society of Petroleum Engineers, Dallas, 1999.
9. J.M. Evans, M.J. Fear, and N.C. Meany, "A New Graphical Representation for Rule Definition and Explanation in an Expert System," *Applications and Innovations in Intelligent Systems III*, Springer-Verlag, Berlin, 1995.
10. R. Kremer, "Concept Mapping Tool to Handle Multiple Formalisms," *AAAI Spring Symp. Artificial Intelligence in Knowledge Management*, AAAI Press, Menlo Park, Calif., 1997.
11. R. MacGregor, "The Evolving Technology of Classification-Based Knowledge Representation Systems," *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, Morgan Kaufmann, San Francisco, 1991, pp. 385–400.
12. J-L. Ermine, "Knowledge Management in the Commissariat a l'Energie Atomique," *PAKeM98—Practical Application of Knowledge Management*, Practical Applications Co., London, 1998.
13. W.J. Clancey, "Model Construction Operators," *Artificial Intelligence*, vol. 53, 1992, pp. 1–115.
14. R.J. Brachman et al., "Living with Classic: When and How to Use a KL-ONE-like Language," *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, Morgan Kaufmann, San Francisco, 1991, pp 401–456.
15. S. White, *Enhancing Knowledge Acquisition with Constraint Technology*, doctoral dissertation, Dept. Computing Science, University of Aberdeen, Aberdeen, UK, 2000.